

Sem For		Strings (cont)		Classes		Encontre o que procura	
array.slice(inicio, fim?)	Cria uma nova lista	Eu sou uma \${variavelQualquer}	Interpolação de string	class Oi {}	cria uma nova classe Oi	array.find(fn(item))	Retorna o primeiro item encontrado ou undefined
array.map(fn(item, index))	Aplica a função fn para todos os itens e retorna algo	str + str 2	Junta as duas strings	new Oi()	Instancia a classe Oi	array.indexOf(fn(item))	Retorna o indice do primeiro item encontrado ou -1
array.filter(fn(item))	Filtra a array. Fn deve retornar true para o item que devera ser filtrado	const str = "ola"	armazena a string ola dentro da constante str	constructor {}	Pode-se ser colocado dentro da classe para ter um construtor	array.includes(item)	Retornar True ou False
array.reduce(fn(accumulator, current), valorInicial)	Para cada item na lista, a função é aplicada, retornando ao fim o valor total do accumulator	const str = "ola '1' "	armazena a string ola '1' dentro da constante str	Oi.metodo()	chama um novo metodo	Regex no javascript	
array.forEach(fn(item, index))	Aplica a função fn para cada item do array	const str = 'ola "1" '	armazena a string ola "1" dentro da constante str	class ClassWithPrivateField { #privateField; }	Cria uma classe com um campo privado. Serve para metodos ou atributos. Só precisa colocar # antes	const regex = /novoRegex/gmi	Cria uma nova expressão regex que pode ser usada posteriormente
array.join(delimitador)	Transforma a array em uma string	\	Caracter de escape	Coisas legais		str.replace(regex, novoValor)	Substitui o que foi encontrado na string pelo novoValor
Strings		Funções		const newObj = {...oldObject}	Cria um novo objeto igual ao anterior	regex.test(str)	Retorna verdadeiro caso de um match ou falso caso não
str.split(delimitador)	Retorna uma lista contendo os elementos da string separados pelo demilitador	(parametro1, parametro2) => {}	Cria uma nova função	const newList = [...oldList]	Cria uma lista igual a anterior	regex.exec(string)	Retorna null em caso de não encontrar nada, retorna uma lista dos matches e grupos de captura
str.toUpperCase()	Retorna a string com todos os caracteres maiusculos	function(parametro1, parametro2) {}	Cria uma nova função	const [primeiro, segundo, terceiro] = [1, 2, 3]	Desconstro a lista em três constantes	str.match(regex)	Retorna uma lista com todos os itens (Não suporta grupo de captura)
str.toLowerCase()	Retorna a string com todos os caracteres minusculos	() => {{key: valor}}	cria uma arrow function de uma linha que retorna um objeto	const [primeiro, segundo, terceiro] = obj	Desconstro um obj para as constantes. O Objeto precisa ter as chaves iguais para funcionar!	str.matchAll(regex)	Retorna um iterator que pode ser usado com for de todos os matches(Suporta lista de captura)
		() => "ola"	Cria uma arrow function que retorna a string ola. Caso a arrow function possua apenas uma linha, não precisa de {}				



Regex no javascript (cont)

[...str.matchAll(regex)]
Lista de todos os matches (suporta grupo de captura)

Objetos

const obj = {}
Criando um novo objeto

const obj = {chave: 'valorQualquer'}
Criar objeto com valor inicial

obj.chaveUnica
Acessar o valor de um objeto

delete obj.chaveUnica
Deleta uma chave

Object.keys(obj)
Retornar um array de todas as chaves de um elemento

Object.keys(obj),map(key => obj[key])
Acessando todos os valores de um objeto

Object.values(obj)
Retornar um array de todos os valores de um objeto

Object.assign({}, ...listaDeObjetos)
Cria um novo object mesclando todos os objetos da lista

Object.freeze(obj);
Torna um objeto imutável

Objetos são como dicionários (python). Eles possuem uma chave única (key) e um valor (Qualquer coisa). Podemos criar um objeto com a seguinte sintaxe:

Listas

array.push(novoItem)
Adiciona um novo item ao array

array.pop()
Remove o último elemento do array

array.shift()
Remove o primeiro elemento do array

array.unshift(item)
Adiciona um item ao inicio do array

array.splice(indexParaRemover, numeroDeItemsParaRemover)
Remove um item de determinada posição de um array

fruits.slice()
Copia o array

Promessas

promise.then(function resolve(res))
Caso de sucesso, a função dentro do then sera executada. Em de erro, a função dentro do catch sera executada

async function fn(){}

await promise
Faz com que todo o código abaixo do await espere para ser executado. Por baixo dos panos, coloca tudo que esta abaixo dentro de um then.

Promessas (cont)

Promise.all(listaDePromessas)
Executa todas as promises em paralelo e só executa o código abaixo quando todas são executadas e bem-sucedidas. Retorna uma lista com o resultado das promessas

Promise.allSettled(listaDePromessas)
Executa todas as promises em paralelo e só executa o código abaixo quando todas são executadas, com ou sem erro. Retorna uma lista com o resultado das promessas

new Promise(resolve, reject) => {}
Cria uma nova promessa. Chame resolve e passe o resultado em caso de sucesso. Chame o reject em caso de falha.

Uma promessa é um código assincrono, que sera mandado ao event loop. O resultado só estara disponível depois, mas a aplicação continuara executando o código. Ele é não bloqueante